The Wisconsin PASS Project

Remzi Arpaci-Dusseau
University of Wisconsin, Madison

Trends: The Three C's

Copious Data

- Massive needs for many interesting applications

Complexity

- Each component of a scalable system -> comple

Commodity

- Use commodity PCs, OS's, disks

Implications

Copious Data

- Need to build large, scalable systems

Complexity

- How to understand what system is doing?

Commodity

- Cheap parts -> unreliable parts

Goal: Build scalable systems that work “well” despite component complexity and use of commodity parts

Key: Fault Handling

Underlying challenge:
How system handles faults

Why?

- Massive data -> massive systems ->
  faults are commonplace
- Complexity often dominated by fault handling
  Hoare: “Unavoidable price of reliability is simplici
- Cheap components -> Less tested, less reliable

How Do Systems Handle Faults?

Previous work says: "Not well"

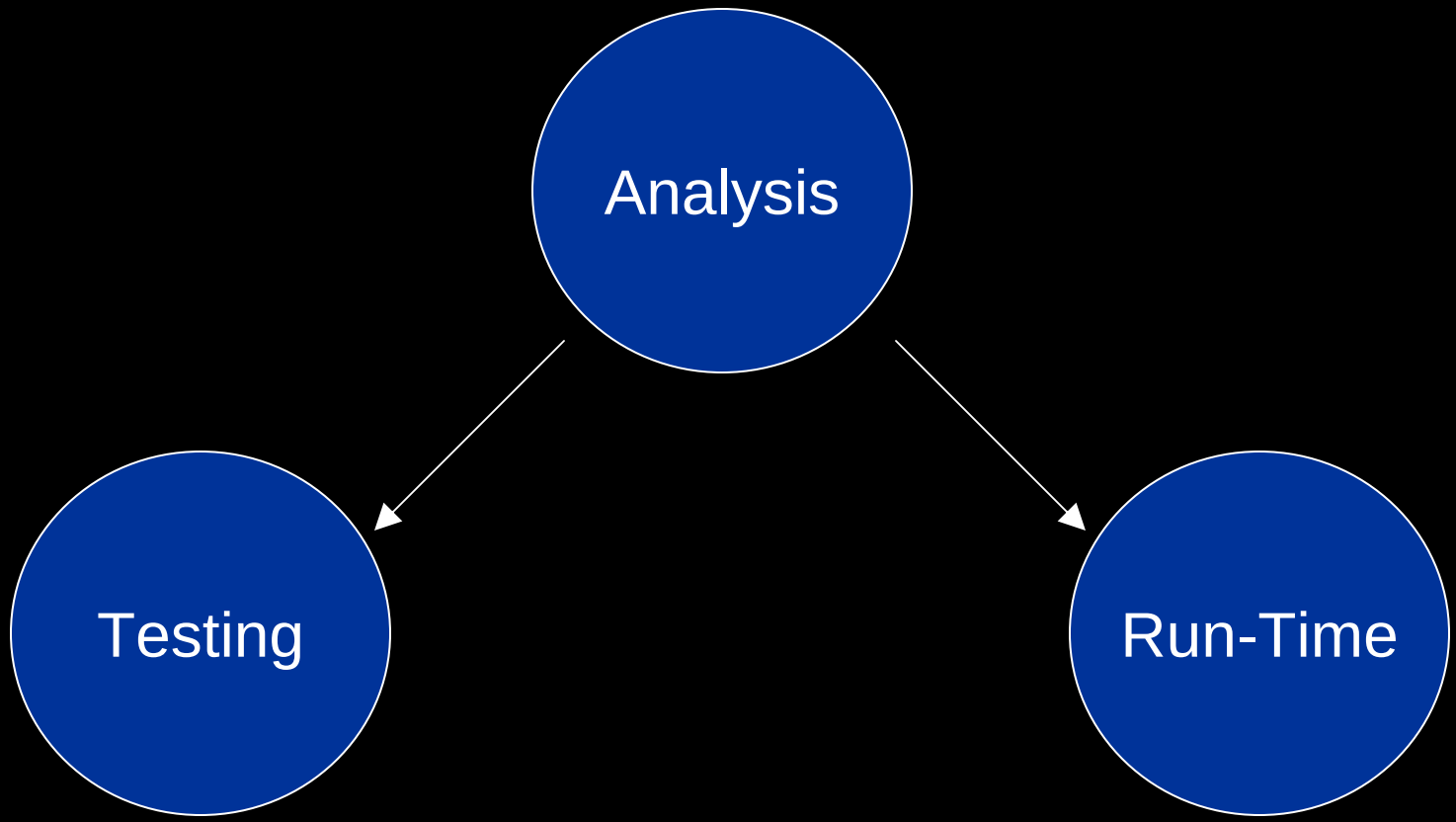
- Analysis of local Linux file systems [SOSP '05]: fault-handling policies are poorly specified
- Examples: oddly inconsistent, often buggy

How discovered?

- Testing framework:
  Inject faults, see how system reacts
- But source-oblivious testing has its limits...

In This Proposal

The Wisconsin PASS Project
Program Analysis of Storage Systems



PASS Details

Analysis

- Use PL techniques to understand fault-handling policy
- e.g., how are errors propagated through a file system?

Testing

- Use analysis to drive better testing
- e.g., does fault injection cover fault-handling code?

Run-time

- Instrumentation to gather failure data
- e.g., what kind of failures occur in the field?

Research Team

"Micro-interdisciplinary" approach

- Andrea Arpaci-Dusseau
- Remzi Arpaci-Dusseau
- Ben Liblit
- Mike Swift
- Miron Livny

Storage

PL

Device Drivers

Condor

Platforms

Local File Systems

- Start with simpler systems: ext3, ReiserFS, etc.
- Extend to entire stack: drivers, etc.

Beyond local: Distributed

- Lustre or commercial scalable system (e.g., from EMC or NetApp)

Wrap up

Storage systems are getting...

- Big
- Complex
- Comprised of commodity pieces

Failure will be common

Key to building robust, reliable, scalable systems
Use more sophisticated analysis to drive understanding, testing, and instrumentation